Bubble and Selection Sorts

Objectives

- In this Week, you will learn to:
  - Identify the algorithms that can be used to sort data
  - Sort data by using bubble sort
  - Sort data by using selection sort

Sorting Data

- Sorting is the process of arranging data in some pre-defined order or sequence. The order can be either ascending or descending.
- If the data is ordered, you can directly go to the section , thereby reducing the number of records to be traversed.

Selecting a Sorting Algorithm

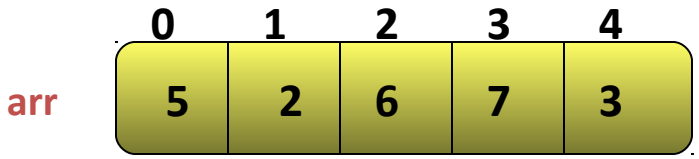
- Sorting is implemented in a program by using an algorithm.
- Some sorting algorithms are:
  - Bubble sort
  - Selection sort
  - Insertion sort
  - Shell sort
  - Merge sort
  - Quick sort

Sorting data by Using Bubble Sort

- Bubble sort algorithm:
  - Is one of the simplest sorting algorithms
  - Has a quadratic order of growth and is therefore suitable for sorting small lists only
  - Works by repeatedly scanning through the list, comparing adjacent elements, and swapping them if they are in the wrong order

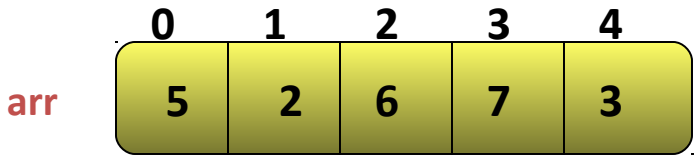
Implementing Bubble Sort Algorithm

- To understand the implementation of bubble sort algorithm, consider an unsorted list of numbers stored in an array.



Implementing Bubble Sort Algorithm (Contd.)

- Let us sort this unsorted list.

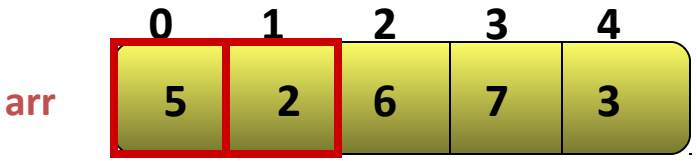


Implementing Bubble Sort Algorithm (Contd.)

Pass 1

n = 5

- Compare the element stored at index 0 with the element stored at index 1.

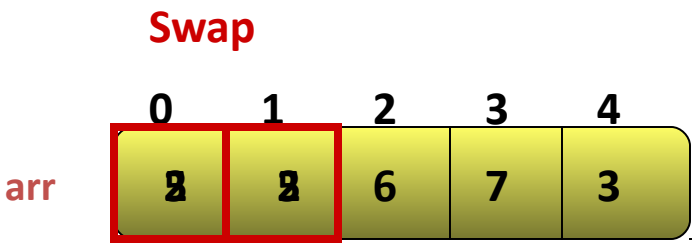


Implementing Bubble Sort Algorithm (Contd.)

Pass 1

n = 5

- Swap the values if they are not in the correct order.



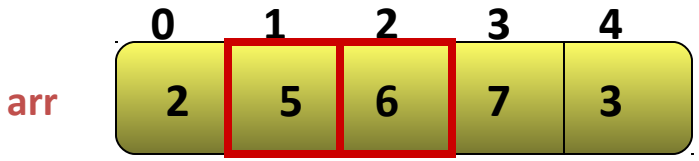
Implementing Bubble Sort Algorithm (Contd.)

Pass 1

n = 5

- Compare the element stored at index 1 with the element stored at index 2 and swap the values if the value at index 1 is greater than the value at index 2.

No Change



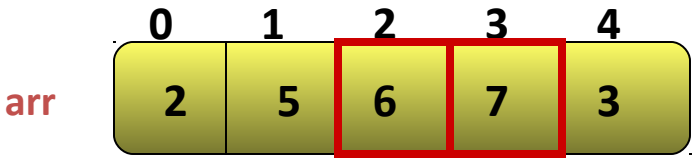
Implementing Bubble Sort Algorithm (Contd.)

Pass 1

n = 5

- Compare the element stored at index 2 with the element stored at index 3 and swap the values if the value at index 2 is greater than the value at index 3.

No Change

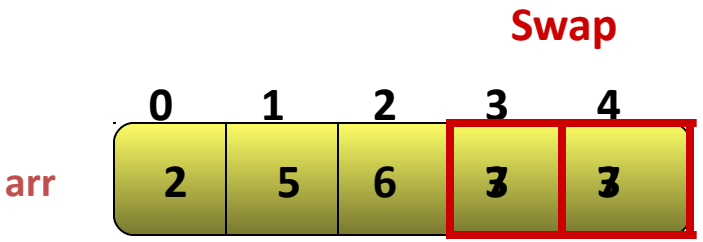


Implementing Bubble Sort Algorithm (Contd.)

Pass 1

n = 5

- Compare the element stored at index 3 with the element stored at index 4 and swap the values if the value at index 4 is greater than the value at index 4.

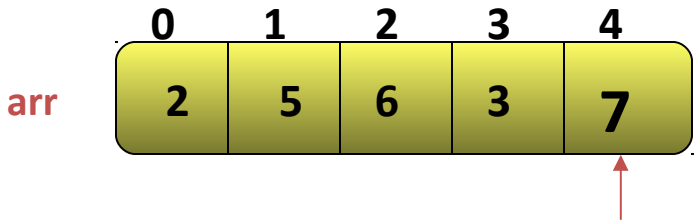


Implementing Bubble Sort Algorithm (Contd.)

Pass 1

n = 5

- Compare the element stored at index 3 with the element stored at index 4 and swap the values if the value at index 4 is greater than the value at index 4.



Largest element is placed at its correct position after Pass 1

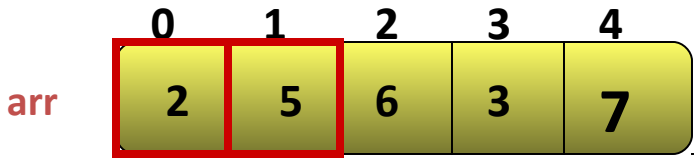
Implementing Bubble Sort Algorithm (Contd.)

Pass 2

n = 5

- Compare the element stored at index 0 with the element stored at index 1 and swap the values if the value at index 0 is greater than the value at index 1.

No Change



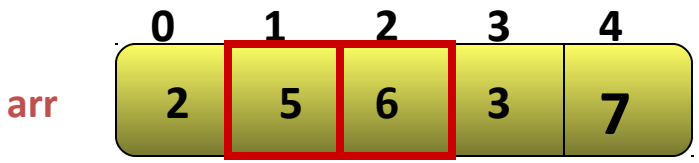
Implementing Bubble Sort Algorithm (Contd.)

Pass 2

n = 5

- Compare the element stored at index 1 with the element stored at index 2 and swap the values if the value at index 1 is greater than the value at index 2.

No Change

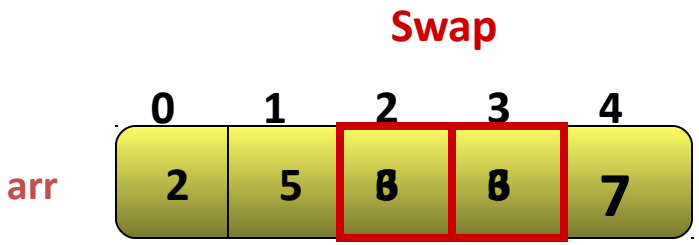


Implementing Bubble Sort Algorithm (Contd.)

Pass 2

n = 5

- Compare the element stored at index 2 with the element stored at index 3 and swap the values if the value at index 2 is greater than the value at index 3.

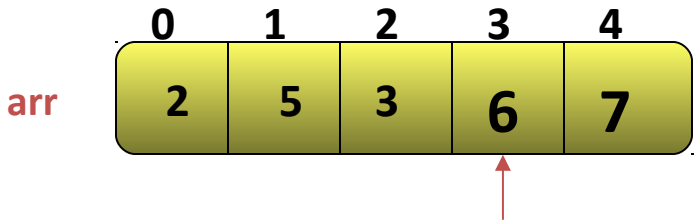


Implementing Bubble Sort Algorithm (Contd.)

Pass 2

n = 5

- Compare the element stored at index 2 with the element stored at index 3 and swap the values if the value at index 2 is greater than the value at index 3.



Second largest element is placed at its correct position after Pass 2

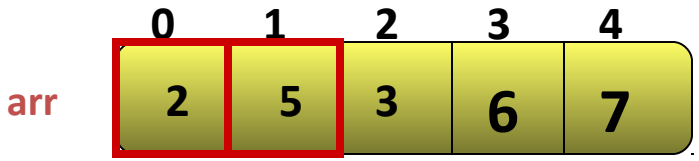
Implementing Bubble Sort Algorithm (Contd.)

Pass 3

n = 5

- Compare the element stored at index 0 with the element stored at index 1 and swap the values if the value at index 1 is greater than the value at index 1.

No Change

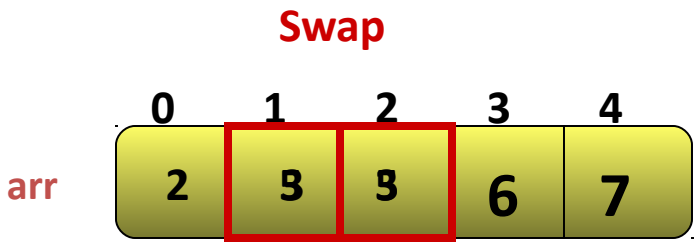


Implementing Bubble Sort Algorithm (Contd.)

Pass 3

n = 5

- Compare the element stored at index 1 with the element stored at index 2 and swap the values if the value at index 1 is greater than the value at index 2.

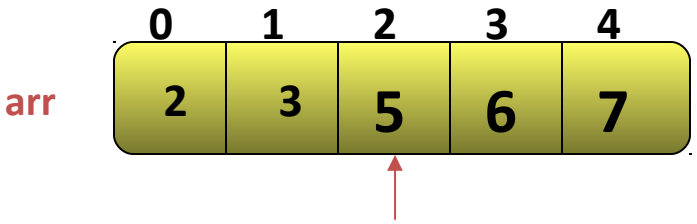


Implementing Bubble Sort Algorithm (Contd.)

Pass 3

n = 5

- Compare the element stored at index 2 with the element stored at index 3 and swap the values if the value at index 2 is greater than the value at index 3.



Third largest element is placed at its correct position after Pass 3

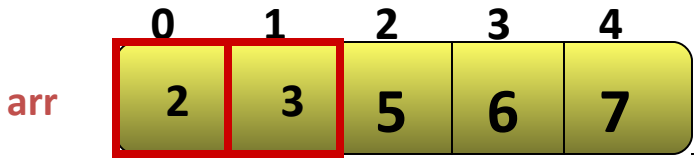
Implementing Bubble Sort Algorithm (Contd.)

Pass 4

n = 5

- Compare the element stored at index 0 with the element stored at index 1 and swap the values if the value at index 0 is greater than the value at index 1.

No Change

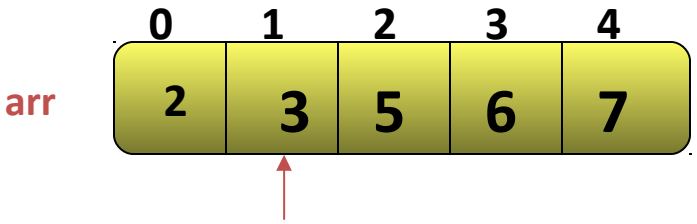


Implementing Bubble Sort Algorithm (Contd.)

Pass 4

n = 5

- Compare the element stored at index 0 with the element stored at index 1 and swap the values if the value at index 0 is greater than the value at index 1.



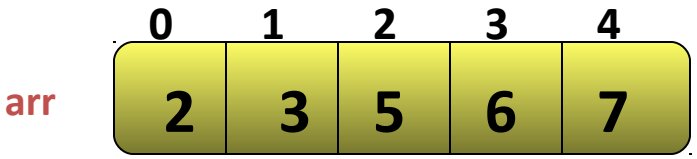
Fourth largest element is placed at its correct position after Pass 4

Implementing Bubble Sort Algorithm (Contd.)

Pass 4

n = 5

- At the end of Pass 4, the elements are sorted.



Implementing Bubble Sort Algorithm (Contd.)

```
Algorithm BubbleSort ( A [],n )
  {

    1. for ( pass = 1 to n-1 )
          1.1 for ( j= 0 to n – 1 – pass )
                1.1.1   If ( A [j] > A [j+1] )
                        1.1.1.1   t = A [j]
                        1.1.1.2   A [j] = A [j+1]
                        1.1.1.3   A [j+1] = t
     }
```

Determining the Efficiency of Bubble Sort Algorithm

- The efficiency of a sorting algorithm is measured in terms of number of comparisons.
- In bubble sort, there are n – 1 comparisons in Pass 1, n – 2 comparisons in Pass 2, and so on.
- Total number of comparisons = (n – 1) + (n – 2) + (n – 3) + … + 3 + 2 + 1 = n(n – 1)/2.
- n(n – 1)/2 is of $O(n^2)$ order. Therefore, the bubble sort algorithm is of the order $O(n^2)$.

Just a minute

- What is the order of growth of the bubble sort algorithm?

- Answer:
  - The bubble sort algorithm has a quadratic order of growth.

Just a minute

- While implementing bubble sort algorithm, how many comparisons will be performed in Pass 1.

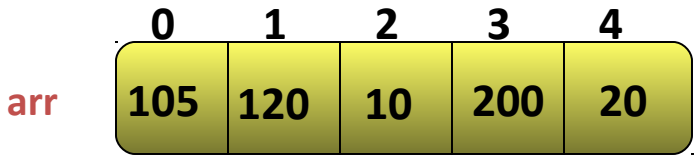
- Answer:
  - n – 1 comparisons

Sorting Data by Using Selection Sort

- Selection sort algorithm:
  - Has a quadratic order of growth and is therefore suitable for sorting small lists only
  - Scans through the list iteratively, selects one item in each scan, and moves the item to its correct position in the list

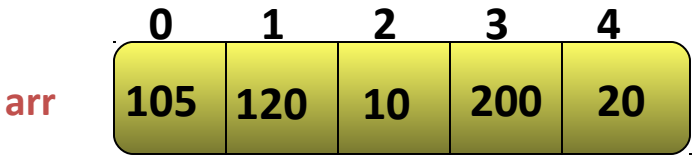
Implementing Selection Sort Algorithm

- To understand the implementation of selection sort algorithm, consider an unsorted list of numbers stored in an array.



Implementing Selection Sort Algorithm (Contd.)

- Let us sort this unsorted list.

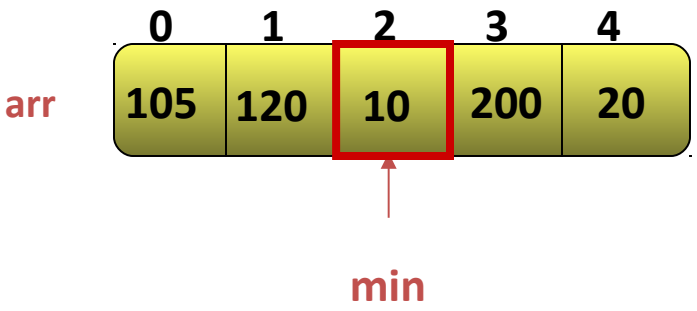


Implementing Selection Sort Algorithm (Contd.)

Pass 1

n = 5

- Search the minimum value in the array, arr[0] to arr[n – 1].

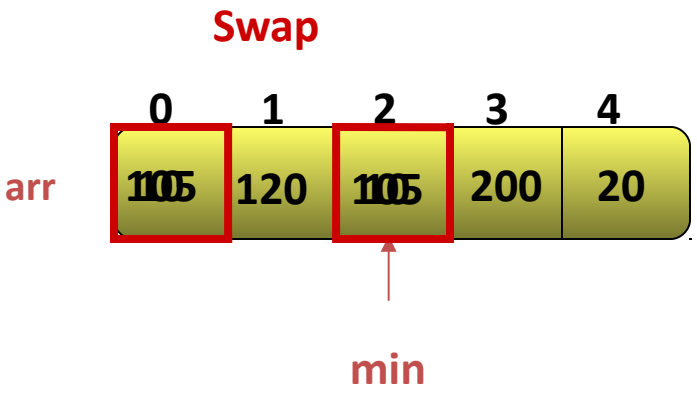


Implementing Selection Sort Algorithm (Contd.)

Pass 1

n = 5

- Search the minimum value in the array, arr[0] to arr[n – 1].
- Swap the minimum value with the value at index 0.

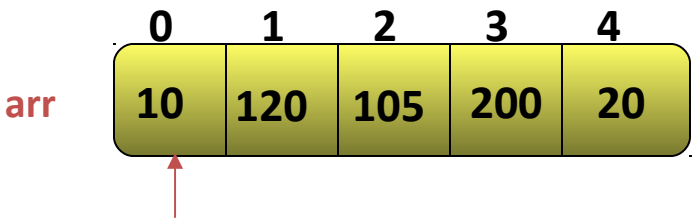


Implementing Selection Sort Algorithm (Contd.)

Pass 1

n = 5

- Search the minimum value in the array, arr[0] to arr[n – 1].
- Swap the minimum value with the value at index 0.



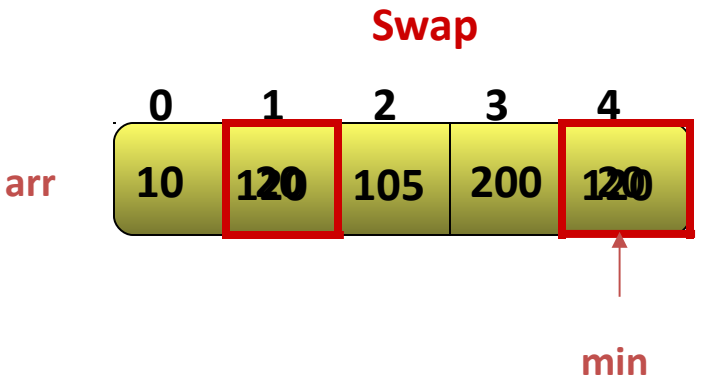
The smallest value is placed at its correct location after Pass 1

Implementing Selection Sort Algorithm (Contd.)

Pass 2

n = 5

- Search the minimum value in the array, arr[1] to arr[n – 1].
- Swap the minimum value with the value at index 1.

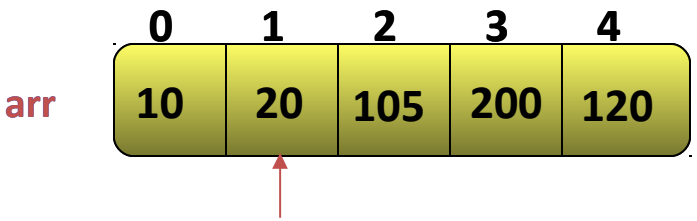


Implementing Selection Sort Algorithm (Contd.)

Pass 2

n = 5

- Search the minimum value in the array, arr[1] to arr[n – 1].
- Swap the minimum value with the value at index 1.



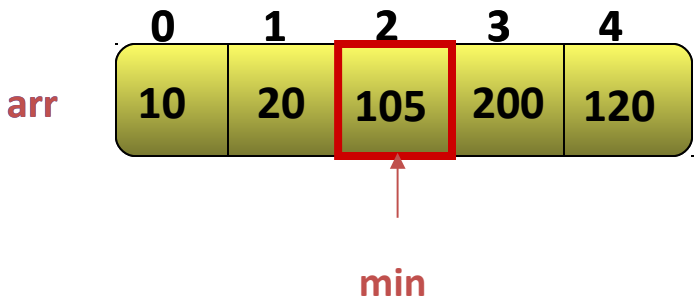
The second smallest value is placed at its correct location after Pass 2

Implementing Selection Sort Algorithm (Contd.)

Pass 3

n = 5

- Search the minimum value in the array, arr[2] to arr[n – 1].
- Swap the minimum value with the value at index 2.

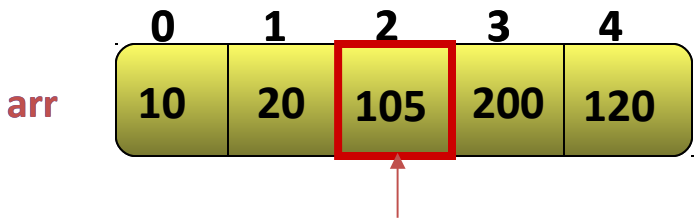


Implementing Selection Sort Algorithm (Contd.)

Pass 3

n = 5

- Search the minimum value in the array, arr[2] to arr[n – 1].
- Swap the minimum value with the value at index 2.



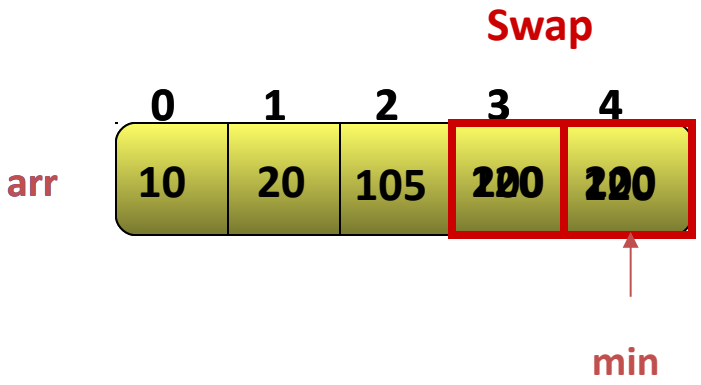
The third smallest value is placed at its correct location after Pass 3

Implementing Selection Sort Algorithm (Contd.)

Pass 4

n = 5

- Search the minimum value in the array, arr[3] to arr[n – 1].
- Swap the minimum value with the value at index 3.

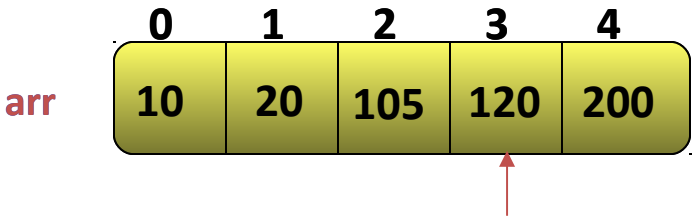


Implementing Selection Sort Algorithm (Contd.)

Pass 4

n = 5

- Search the minimum value in the array, arr[3] to arr[n – 1].
- Swap the minimum value with the value at index 3.



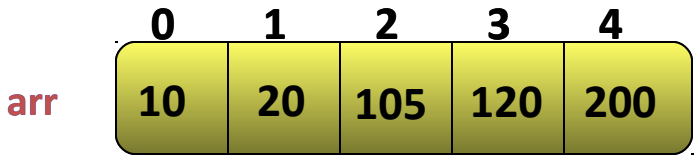
The fourth smallest value is placed at its correct location after Pass 4

Implementing Selection Sort Algorithm (Contd.)

Pass 4

n = 5

- The list is now sorted.



Implementing Selection Sort Algorithm (Contd.)

```
Algorithm SelectionSort ( A [],n )
{
        1. for( i = 0 to n-2 )
                        1.1 Set min  =  i
                      1.2 for ( j= i+1 to n-1 )
                                    1.2.1 if ( A [j] < A [min] )
                                    1.2.1.1 min = j

                        1.3 swap A [i] and A [min]

    }
```

Determining the Efficiency of Selection Sort Algorithm

- In selection sort, there are n – 1 comparisons during Pass 1 to find the smallest element, n – 2 comparisons during Pass 2 to find the second smallest element, and so on.
- Total number of comparisons = (n – 1) + (n – 2) + (n – 3) + … + 3 + 2 + 1 = n(n – 1)/2
- n(n – 1)/2 is of $O(n^2)$ order. Therefore, the selection sort algorithm is of the order $O(n^2)$.

Just a minute

- Read the following statement and specify whether it is true or false:
  - The efficiency of selection sort algorithm is same as that of the bubble sort algorithm.

- Answer:
  - True

Just a minute

- How many comparisons are performed in the second pass of the selection sort algorithm?

- Answer:
  - n – 2